3 Functors and natural transformations

3.1 Functors

A functor is a morphism between categories, the standard example is the List functor which is an endofunctor on the category of sets, i.e. it is a functor from $\underline{\mathbf{Set}}$ to $\underline{\mathbf{Set}}$, I write List : $\underline{\mathbf{Set}} \to \underline{\mathbf{Set}}$ overloading $\to$[6]. The list functor has an effect on objects, i.e. sets, given $A : \mathbf{Set}$ we obtain $\mathrm{List}\, A : \mathbf{Set}$ which is the set of lists over A which we write as $[a_0, a_1, \ldots, a_{n-1}]$, this includes the empty list $[]$. The List functor comes with a function which "maps" a function $f : A \to B$ to $\mathrm{List}\, f : \mathrm{List}\, A \to \mathrm{List}\, B$, which is defined as

$$\mathrm{List}\, f\, [a_0, a_1, \ldots, a_{n-1}] = [f\, a_0, f\, a_1, \ldots, f\, a_{n-1}]$$

Note that we overload List to mean at the same time a map on objects and a map on morphisms. List also preserves the categorical structure, it maps identity to identity, that is $\mathrm{List}\, \mathrm{id}_A = \mathrm{id}_{\mathrm{List}\, A}$ and $\mathrm{List}\, (f \circ g) = (\mathrm{List}\, f) \circ (\mathrm{List}\, g)$. This can be easily seen by showing that both sides are pointwise equal when applied to a generic list

$$\begin{aligned} \mathrm{List}\, \mathrm{id}_A\, [a_0, a_1, \ldots, a_{n-1}] &= [\mathrm{id}\, a_0, \mathrm{id}\, a_1, \ldots, \mathrm{id}\, a_{n-1}] \\ &= [a_0, a_1, \ldots, a_{n-1}] \\ &= \mathrm{id}_{\mathrm{List}\, A}\, [a_0, a_1, \ldots, a_{n-1}] \end{aligned}$$

and

$$\begin{aligned} \mathrm{List}\, (f \circ g)\, [a_0, a_1, \ldots, a_{n-1}] &= [(f \circ g)\, a_0, (f \circ g)\, a_1, \ldots, (f \circ g)\, a_{n-1}] \\ &= [f\, (g\, a_0), f\, (g\, a_1), \ldots, f\, (g\, a_{n-1})] \\ &= \mathrm{List}\, f\, [g\, a_0, g\, a_1, \ldots, g\, a_{n-1}] \\ &= \mathrm{List}\, f\, (\mathrm{List}\, g\, [a_0, a_1, \ldots, a_{n-1}]) \\ &= ((\mathrm{List}\, f) \circ (\mathrm{List}\, g))\, [a_0, a_1, \ldots, a_{n-1}] \end{aligned}$$

More formally the proofs proceed by list induction.

Inspired by List we define the notion of a functor in general: given categories $\underline{\mathbf{C}}$ and $\underline{\mathbf{D}}$ a functor $F : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ is given by a map on objects $F : |\underline{\mathbf{C}}| \to |\underline{\mathbf{D}}|$ and a map on morphisms $F : \underline{\mathbf{C}}(A, B) \to \underline{\mathbf{D}}(F\, A, F\, B)$ that preserves identities $F\, \mathrm{id}_A = \mathrm{id}_{F\, A}$ and composition $F\, (f \circ g) = (F\, f) \circ (F\, g)$.

Exercise 13 *Show that every functor preserves isomorphisms. That is if $\phi : A \to B$ is an isomorphism then so is $F\, \phi : F\, A \to F\, B$.*

We note that functors between preorder categories are simply monotone maps, i.e. correspond to morphisms in $\underline{\mathbf{Pre}}$ and functors between monoids are monoid morphisms, i.e. correspond to morphisms in $\underline{\mathbf{Mon}}$. Given these observations above it may appear a good idea to define a category of categories

[6] In the literature people often write $[\underline{\mathbf{Set}}, \underline{\mathbf{Set}}]$.

with functors as morphisms. You find this definition in many classical text books. We will not do this because it isn't clear what equality of functors is, they do not form a set in our sense. [7]

Categories form another structure, a higher category. Functors between two categories don't form a set they form a category themselves. The morphisms between functors are called *natural transformations* and in the case of endofunctors on sets they correspond to a well known concept from functional programming: polymorphic functions.

3.2 Natural transformations

Let's look at an example: the reverse function on lists. Given a set A we have a function $\mathrm{rev}_A : \mathrm{List}\, A \to \mathrm{List}\, A$ which reverses its input, i.e.

$$\mathrm{rev}_A\, [a_0, a_1, \ldots, a_{n-1}] = [a_{n-1}, \ldots, a_1, a_0]$$

This is a family of functions, but using a bit of type-theoretic mumbo jumbo we can view it as one function $\mathrm{rev} : \Pi_{A:\mathbf{Set}} \mathrm{List}\, A \to \mathrm{List}\, A$ and we agree to omit the set parameter (or put it in subscript).

This function has an important property it is *natural*, which means that it commutes with the map operation on lists, that is given any function $f : A \to B$

$$\begin{array}{ccc} \mathrm{List}\, A & \xrightarrow{\mathrm{rev}_A} & \mathrm{List}\, A \\ {\scriptstyle \mathrm{List}\, f}\downarrow & & \downarrow{\scriptstyle \mathrm{List}\, f} \\ \mathrm{List}\, B & \xrightarrow[\mathrm{rev}_B]{} & \mathrm{List}\, B \end{array}$$

Let's check that this diagram commutes by applying both paths from the top left to the bottom right to a generic list:

$$\begin{aligned} (\mathrm{rev}_B \circ \mathrm{List}\, f)\, [a_0, a_1, \ldots, a_{n-1}] &= \mathrm{rev}_B\, (\mathrm{List}\, f\, [a_0, a_1, \ldots, a_{n-1}]) \\ &= \mathrm{rev}_B\, [f\, a_0, f\, a_1, \ldots, f\, a_{n-1}]) \\ &= [f\, a_{n-1}, \ldots, f\, a_1, f\, a_0]) \\ &= \mathrm{List}\, f\, [a_{n-1}, \ldots, a_1, a_0] \\ &= \mathrm{List}\, f\, (\mathrm{rev}_A\, [a_0, a_1, \ldots, a_{n-1}]) \\ &= (\mathrm{List}\, f \circ \mathrm{rev}_A)\, [a_0, a_1, \ldots, a_{n-1}] \end{aligned}$$

and we conclude that $\mathrm{rev}_B \circ \mathrm{List}\, f = \mathrm{List}\, f \circ \mathrm{rev}_A$. As before formally the proof uses list induction to get rid of the ....

Naturality actually corresponds to the idea of *parametricity* in functional programming: rev is defined uniformly for all sets. It is good to look at a counterexample but actually we cannot define one because in type theory and in functional programming all functions are parametric and hence natural. To

[7] We can however, define the category of strict categories, i.e. categories whose objects form a set. Because in this case equality of functors is a proposition and hence functors form a set.

get our hands on an *unnatural* function let us allow for the moment to define a weird function which analyses a set, that is we define

$$\text{weird} : \Pi_{A:\mathbf{Set}} \text{List}\, A \to \text{List}\, A$$

by

$$\begin{aligned} \text{weird}_{\text{Bool}} &= \text{rev}_{\text{Bool}} \\ \text{weird}_A &= \text{id}_{\text{List}\, A} \quad \text{for all other } A : \mathbf{Set} \end{aligned}$$

That is weird reverses lists of booleans but is the identity every else. weird has the same type as rev hence we have to check the same naturality square. Consider the function even : $\mathbb{N} \to$ Bool which returns true if the input is an even number and false otherwise and apply both sides to the input $[1, 2, 3] : \text{List}\, \mathbb{N}$:

$$\begin{aligned} (\text{weird}_{\text{Bool}} \circ \text{List even})\, [1, 2, 4] & \\ &= \text{weird}_{\text{Bool}}\, (\text{List even})\, [1, 2, 4]) \\ &= \text{weird}_{\text{Bool}}\, ([\text{false}, \text{true}, \text{true}]) \\ &= [\text{true}, \text{true}, \text{false}] \end{aligned}$$

$$\begin{aligned} (\text{List even} \circ \text{weird}_{\mathbb{N}})\, [1, 2, 4] & \\ &= \text{List}\, f\, (\text{weird}_{\mathbb{N}}\, [1, 2, 4]) \\ &= \text{List}\, f\, [1, 2, 4] \\ &= [\text{false}, \text{true}, \text{true}] \end{aligned}$$

Hence, clearly the diagram doesn't commute since $[\text{true}, \text{true}, \text{false}] \neq [\text{false}, \text{true}, \text{true}]$.

Here is some notation for the type of natural transformations. As we use Π in type theory we use an integral in category theory, that is we write [8]

$$\text{rev} : \int_{A:\mathbf{Set}} \text{List}\, A \to \text{List}\, A$$

The integral here is a general concept called *an end* (see subsection 3.5 for the general definition of ends). There are also coends (corresponding to Σ-types in type theory) which also use the integral notation but the index is on the top.

We define in general, what a *natural transformation* is: Given two functors $F, G : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ a natural transformation α from F to G is given by a family of morphisms:

$$\alpha : \Pi_{A:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(F\, A, G\, A)$$

such that the following diagram commutes:

$$\begin{array}{ccc} F\, A & \xrightarrow{\alpha_A} & G\, A \\ {\scriptstyle F\, f}\downarrow & & \downarrow{\scriptstyle G\, f} \\ F\, B & \xrightarrow[\alpha_B]{} & F\, B \end{array}$$

[8] Just think of *parametric* Π *type* when you see the integral.

To express that α is a natural transformation we write:

$$\alpha : \int_{A:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(F\,A, G\,A)$$

Exercise 14

1. *Show that* $\mathrm{Maybe} : \mathbf{Set} \to \mathbf{Set}$ *where* $\mathrm{Maybe}\,A$ *is given by the constructors:*

$$\begin{array}{ll} \mathrm{nothing} & : \mathrm{Maybe}\,A \\ \mathrm{just} & : A \to \mathrm{Maybe}\,A \end{array}$$

   is a functor.

2. *Show that the function* $\mathrm{hd} : \Pi_{A:\mathbf{Set}} \mathrm{List}\,A \to \mathrm{Maybe}\,A$ *which is defined as*

$$\begin{array}{ll} \mathrm{hd}\,[] & = \mathrm{nothing} \\ \mathrm{hd}\,[a_0, a_1, \ldots, a_{n-1}] & = a_0 \end{array}$$

   is a natural transformation (i.e. $\mathrm{hd} : \int_{A:\mathbf{Set}} \mathrm{List}\,A \to \mathrm{Maybe}\,A$*).*

Functors between any two categories $\underline{\mathbf{C}}, \underline{\mathbf{D}}$ and natural transformations form a category, the functor category $\underline{\mathbf{C}} \to \underline{\mathbf{D}}$. It's objects are functors $F : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ and given two functors F, G the homset is the set [9] of natural transformations $\int_{X:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(F\,X, G\,X)$, the identity natural transformation $\mathrm{id} : \int_{X:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(F\,X, F\,X)$ is given by $\mathrm{id}_X = \mathrm{id}_{F\,X}$ and given $\alpha : \int_{X:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(F\,X, G\,X), \beta : \int_{X:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(G\,X, H\,X)$ their composition is

$$\begin{array}{l} (\beta \circ \alpha) : \int_{X:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(F\,X, H\,X) \\ (\beta \circ \alpha)_X = \beta_X \circ \alpha_X \end{array}$$

Exercise 15 *Check that these are indeed natural transformations and that the laws for a category hold.*

3.3 Adjunctions

An important class of functors are the forgetful functors which just *forget* some property or structure; they are usually denoted by U, for example $U : \underline{\mathbf{Mon}} \to \underline{\mathbf{Set}}$ which assigns to any monoid the carrier of the monoid, i.e. $U\,(A, e, \_*\_) = A$. Since functions between monoids are given by functions on the carrier that

[9] This is a set in our sense because two natural transformations are equal when they are pointwise equal (and we don't need to talk about equality of objects). However, they are not necessarily small. The natural transformations between two endofunctors on $\underline{\mathbf{Set_i}}$ are not in $\mathbf{Set}_i$ because we quantify over all small sets.

preserve the structure the morphism part of the functor and the functor laws are straightforward.

Is there an interesting functor in the other direction, that is $F : \underline{\mathbf{Set}} \to \underline{\mathbf{Mon}}$? Indeed there is: we can use the fact that lists form a monoid with append $\_ +\!\!+ \_ : \mathrm{List}\, A \to \mathrm{List}\, A \to \mathrm{List}\, A$ and $[] : \mathrm{List}\, A$, hence on objects we can define

$$F\, A = (\mathrm{List}\, A, [], \_ +\!\!+ \_)$$

We have already observed that List is a functor but to show that F is a functor we also need that $\mathrm{List}\, f$ is a monoid morphism:

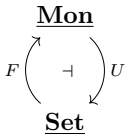
Exercise 16 *Show that for any $f : A \to B$, $\mathrm{List}\, f : \mathrm{List}\, A \to \mathrm{List}\, B$ is a monoid morphism, that is*

$$\mathrm{List}\, f\, [] = []$$
$$\mathrm{List}\, f\, (l +\!\!+ m) = (\mathrm{List}\, f\, l) +\!\!+ (\mathrm{List}\, f\, m)$$

Hence we can define $F\, f = \mathrm{List}\, f$, the functor laws follow because List is a functor on $\underline{\mathbf{Set}}$.

The monoid constructed with lists is special, it is the *free monoid* over a given set. Here free means that it only satisfies the laws of a monoid but is not subject to any other laws. There are other functors with the type $\underline{\mathbf{Set}} \to \underline{\mathbf{Mon}}$ for example the finite multisets or the finite sets over a given set. These do satisfy additional laws, the counterpart to the append operation is commutative in the case of multisets $l +\!\!+ m = m +\!\!+ l$ and idempotent in the case of finite sets $l +\!\!+ l = l$, hence they are not free over monoids.

How can we say in the language of category theory that a functor is a free functor? We can do this by using the forgetful functor $U : \underline{\mathbf{Mon}} \to \underline{\mathbf{Set}}$. Because the free functor doesn't impose any additional laws there is a (natural) isomorphism between the monoid morphisms between $F\, A$ and a monoid $M = (|M|, e, \_ * \_)$ and A and the underlying set $|M| = U\, M$ of the monoid. We say that the two functors form an adjunction, where F is the left adjoint and U is the right adjoint. We write this as $F \dashv U$ or in a diagram:



Let's go through this in detail, a monoid morphism $f : \underline{\mathbf{Mon}}(F\, A, M)$ is given by a function $f : \mathrm{List}\, A \to |M|$ that preserves the monoid structure. We obtain a function $\phi_{A,M}\, f : A \to |M|$ by just using f on singleton lists:

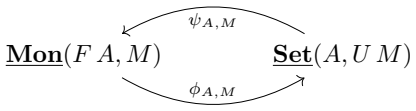
$$\phi\, f\, a = f\, [a]$$

On the other hand given a function $g : A \to |M|$ we can define a function $\psi_{A,M}\, g : \mathrm{List}\, A \to |M|$ using

$$\psi\, g\, [] = e$$
$$\psi\, g\, [a_0, a_1, \ldots, a_{n-1}] = (g\, a_0) * (g\, a_1) * \cdots * (g\, a_n)$$

Exercise 17 *Show that $\psi\, g$ is a monoid morphism, i.e. $\phi\, g : \underline{\mathbf{Mon}}(F\, A, M)$*

Hence we have constructed functions:



Next we show that this is an isomorphism, i.e. ϕ and ψ are inverse to each other. To show that $\phi \circ \psi = \mathrm{id}$ we use extensionality and apply it to an arbitrary $g : A \to |M|$ and $a : A$ and show that $(\phi \circ \psi)\, g\, a = \mathrm{id}\, g\, a$:

$$\begin{aligned}(\phi \circ \psi)\, g\, a &= \phi\, (\psi\, g)\, a \\ &= \psi\, g\, [a] \\ &= g\, a \\ &= (\mathrm{id}\, g)\, a\end{aligned}$$

The other direction is a bit more interesting: to show that $\psi \circ \phi = \mathrm{id}$ we again use extensionality and apply both sides to $f : \mathrm{List}\, A \to |M|$ and an arbitrary $[a_0, a_1, \ldots, a_{n-1}] : \mathrm{List}\, A$ and show that $(\psi \circ \phi)\, f\, [a_0, a_1, \ldots, a_{n-1}] = \mathrm{id}\, f\, [a_0, a_1, \ldots, a_{n-1}]$:

$$\begin{aligned}(\psi \circ \phi)\, f\, [a_0, a_1, \ldots, a_{n-1}] &= \psi\, (\phi\, f)\, [a_0, a_1, \ldots, a_{n-1}] \\ &= (\phi\, f\, a_0) * (\phi\, f\, a_1) * \cdots * (\phi\, f\, [a_n]) \\ &= (f\, [a_0]) * (f\, [a_1]) * \cdots * (f\, a_n) \\ &= f([a_0] +\!\!+ [a_1] +\!\!+ \ldots +\!\!+ [a_n]) \\ &\qquad \text{since } f \text{ is a monoid morphism!} \\ &= f\, [a_0, a_1, \ldots, a_{n-1}] \\ &= (\mathrm{id}\, f)\, [a_0, a_1, \ldots, a_{n-1}]\end{aligned}$$

Exercise 18 *Verify that ϕ is a natural transformation, that is for any f : $\underline{\mathbf{Set}}(B, A)$ and $g : \underline{\mathbf{Mon}}(M, N)$ the following diagram commutes:*

$$\begin{array}{ccc} \underline{\mathbf{Mon}}(F\, A, M) & \xrightarrow{\phi_{A,M}} & \underline{\mathbf{Set}}(A, U\, M) \\ \Big\downarrow{\scriptstyle \lambda h.g \circ h \circ (F\, f)} & & \Big\downarrow{\scriptstyle \lambda k.(U\, g) \circ k \circ f} \\ \underline{\mathbf{Mon}}(F\, B, N) & \xrightarrow[\phi_{B,N}]{} & \underline{\mathbf{Set}}(B, U\, N) \end{array}$$

Question 19 *Why don't we need to verify that ψ is a natural transformation?*

In practice one rarely checks naturality conditions because they follow from the construction.

Question 20 *What goes wrong if we would have used one of the other functors, e.g. finite multisets or finite sets?*

We abstract from the concrete case and define adjunctions in general: Given functors $L : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ and $R : \underline{\mathbf{D}} \to \underline{\mathbf{C}}$ an adjunction is given by a natural isomorphism (in A, B): that is there is a family of isomorphisms

$$\underline{\mathbf{C}}(L\,A, B) \underset{\phi_{A,B}}{\overset{\psi_{A,B}}{\cong}} \underline{\mathbf{D}}(A, R\,B)$$

which is natural in A, B,i.e. f or $f : \underline{\mathbf{D}}(A, C)$ and $g : \underline{\mathbf{C}}(B, D)$ the following diagram commutes

$$\begin{array}{ccc} \underline{\mathbf{C}}(L\,A, B) & \xrightarrow{\phi_{A,B}} & \underline{\mathbf{D}}(A, R\,B) \\ \downarrow{\scriptstyle \lambda h.g \circ h \circ (L\,f)} & & \downarrow{\scriptstyle \lambda k.(R\,g) \circ k \circ f} \\ \underline{\mathbf{C}}(L\,C, D) & \xrightarrow[\phi_{C,D}]{} & \underline{\mathbf{D}}(C, R\,D) \end{array}$$

In this case we say that L is left adjoint to R (or that R is right adjoint to L) and write $L \dashv R$ or in a diagram:

$$\begin{array}{ccc} & \underline{\mathbf{C}} & \\ L \curvearrowright & \dashv & \curvearrowleft R \\ & \underline{\mathbf{D}} & \end{array}$$

We can actually obtain the list functor as the composition of the forgetful and the free functor: List $= F \circ U$. We will return to this observation later, because functors which are obtained by composing a free and a forgetful functor are always monads and indeed List is a monad.

Exercise 21 *There is also a forgetful functor from the category of preorders to the category of endorelations $U : \underline{\mathbf{Pre}} \to \underline{\mathbf{Rel}}$, which is defined just as for preorders but without any conditions, i.e. the objects are sets $X : \mathbf{Set}$ with a relation $R : X \to X \to \mathbf{Prop}$ and morphisms are defined as functions that preserve the relation (as for $\underline{\mathbf{Pre}}$). Now U doesn't do anything but it just forgets that the relation is a preorder. Can you find a left adjoint $F : \underline{\mathbf{Rel}} \to \underline{\mathbf{Pre}}$ which constructs the free preorder over a given relation?*

There are many ways to define adjunctions, the one I have given here is maybe not the best, especially the naturality conditions are quite complicated. A popular alternative is to observe that we can obtain two natural transformations from the natural equivalence, which are called the unit and the counit of

an adjunction:

$$\eta : \int_{A:\underline{\mathbf{D}}} \underline{\mathbf{D}}(A, R\,(L\,A))$$
$$\eta_A = \phi_{A,L\,A}\,\mathrm{id}_{L\,A}$$
$$\epsilon : \int_{B:\underline{\mathbf{C}}} \underline{\mathbf{C}}(L\,(R\,B), B)$$
$$\epsilon_B = \psi_{R\,B,B}\,\mathrm{id}_{R\,B}$$

which make the following diagrams commute (the triangle laws):

$$\begin{array}{ccc} L\,A & \xrightarrow{L\,\eta_A} & L\,(R\,(L\,A)) \\ & \searrow & \downarrow{\scriptstyle \epsilon_{L\,A}} \\ & & L\,A \end{array} \qquad \begin{array}{ccc} R\,B & \xrightarrow{\eta_{R\,B}} & R\,(L\,(R\,B)) \\ & \searrow & \downarrow{\scriptstyle R\,\epsilon_B} \\ & & R\,B \end{array}$$

Exercise 22 *Give explicit definitions of η and ϵ for the adjunction between* **Mon** *and* **Set** *we have defined.*

Exercise 23 *Show that the two definitions of an adjunctions between two functors $L : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ and $R : \underline{\mathbf{D}} \to \underline{\mathbf{C}}$ are equivalent:*

- *An adjunction is given by a natural isomorphism*

$$\underline{\mathbf{C}}(L\,A, B) \cong \underline{\mathbf{D}}(A, R\,B)$$

- *A adjunction is given by two natural transformations:*

$$\eta : \int_{B:\underline{\mathbf{D}}} \underline{\mathbf{D}}(B, R\,(L\,B))$$
$$\epsilon : \int_{A:\underline{\mathbf{C}}} \underline{\mathbf{C}}(F\,(G\,B), B)$$

s.t. $\epsilon_{L\,A} \circ L\,\eta_A = \mathrm{id}_{L\,A}$ and $R\,\epsilon_B \circ \eta_{R\,B} = \mathrm{id}_{R\,B}$.

3.4 The Yoneda lemma

The Yoneda lemma is famous for mystifying the students of category theory, not so much because it is difficult to prove but rather because it is not immediately obvious what it is good for. But then it shows up, maybe unexpectedly again and again, and many constructions proceed with the magic appeal to Yoneda.

Let F be a functor $\underline{\mathbf{C}}^{\mathrm{op}} \to \underline{\mathbf{Set}}$ this is called *a presheaf*[10] . Rather than thinking of presheaves as functors from the dual category, it's useful to treat

[10]This immediately triggers the question: *what is a sheaf?* but the answer is beyond this course. It suffices to say that in order to say what a sheaf is we need a topological structure on the category.

them as *contravariant* functors: on objects, they behave the same as usual functors (sending objects $X : |\underline{\mathbf{C}}|$ to sets $F(X) : \mathbf{Set}$), but on morphisms, they're "flipped around": for $f : \underline{\mathbf{C}}(A, B)$, $F(f)$ is a function from $F(B)$ to $F(A)$.

An example for such a functor is $\underline{\mathbf{C}}(\_, A)$ [11] where $A : |\underline{\mathbf{C}}|$, in this case the functor is called *representable*. The morphism part is given by composition, that is given $f : \underline{\mathbf{C}}(B, C) = \underline{\mathbf{C}^{\mathrm{op}}}(C, B)$:

$$\begin{aligned}&\underline{\mathbf{C}}(f, A) : \underline{\mathbf{C}}(C, A) \to \underline{\mathbf{C}}(B, A)\\&\underline{\mathbf{C}}(f, A)\, g = g \circ f\end{aligned}$$

Exercise 24 *Check that $\underline{\mathbf{C}}(\_, A)$ satisfies the functor laws.*

Indeed, since we already learnt that functors and natural transformations form a category, presheaves over $\underline{\mathbf{C}}$ form a category which we denote as $\mathbf{PSh}\,\underline{\mathbf{C}}$. Now using the other input of the homset of $\underline{\mathbf{C}}$ we can form a functor $\mathbf{Y} : \underline{\mathbf{C}} \to \mathbf{PSh}\,\underline{\mathbf{C}}$, called the *Yoneda embedding*, which is defined on objects as $\mathbf{Y}\, A = \underline{\mathbf{C}}(\_, A)$. To define the morphism part assume $f : \underline{\mathbf{C}}(A, B)$:

$$\begin{aligned}&\mathbf{Y}\, f : \int_{X:\underline{\mathbf{C}}} \underline{\mathbf{C}}(X, A) \to \underline{\mathbf{C}}(X, B)\\&(\mathbf{Y}\, f)_X\, g = f \circ g\end{aligned}$$

Exercise 25 *Check that $Y\, f$ is a natural transformation and that Y satisfies the functor laws.*

Maybe you wondered why presheaves are defined as contravariant functors. The reason is that this way the Yoneda embedding is covariant (otherwise it could be hardly called an embedding).

For any presheaf $F : \underline{\mathbf{C}}^{\mathrm{op}} \to \mathbf{Set}$ we can use its morphism part to define

$$\begin{aligned}\phi_X : F\, X &\to \int_{Y:\underline{\mathbf{C}}} (\mathbf{Y}\, X)\, Y \to F\, Y\\ &: F\, X \to \int_{Y:\underline{\mathbf{C}}} \underline{\mathbf{C}}(Y, X) \to F\, Y\end{aligned}$$

as $\phi_X\, x\, f = F\, f\, x$.

Exercise 26 *Check that ϕ is natural in $X : \underline{\mathbf{C}}$.*

[11] As indicated in the preliminaries we write $\underline{\mathbf{C}}(\_, A)$ for $\lambda X.\underline{\mathbf{C}}(X, A)$. In particular we can apply it, e.g. $\underline{\mathbf{C}}(\_, A)\, B = \underline{\mathbf{C}}(B, A)$.

Now the Yoneda lemma says that this is a natural isomorphism. It is easy to define the inverse (fixing $X : \underline{\mathbf{C}}$):

$$\psi : (\int_{Y:\underline{\mathbf{C}}} \underline{\mathbf{C}}(Y, X) \to F\,Y) \to F\,X$$
$$\psi\,\alpha = \alpha\,\mathrm{id}$$

Now we need to verify that these functions are inverse to each other: We need to prove that $\psi \circ \phi = \mathrm{id}$ which by extensionality means that $(\psi \circ \phi)\,x = x$ where $x : F\,X$

$$\begin{aligned}(\psi \circ \phi)\,x &= (\psi\,(\phi\,x)\\ &= \phi\,x\,\mathrm{id}\\ &= F\,\mathrm{id}\,x\\ &= x\end{aligned}$$

In the other direction we have to show $\phi \circ \psi = \mathrm{id}$ which again by extensionality means that $(\psi \circ \phi)\,\alpha = \alpha$ where $\alpha : \int_{Y:\underline{\mathbf{C}}} \underline{\mathbf{C}}(Y, X) \to F\,Y$ is a natural transformation. Actually we need to go further and apply this to $Y : \underline{\mathbf{C}}$ and $f : \underline{\mathbf{C}}(Y, X)$. If as usual we hide the application to Y using extensionality we need to show that $(\psi \circ \psi)\,\alpha\,f = \alpha\,f$

$$\begin{aligned}(\phi \circ \psi)\,\alpha\,f &= \phi\,(\psi\,\alpha)\,f\\ &= F\,f\,(\psi\,\alpha)\\ &= F\,f\,(\alpha\,\mathrm{id})\end{aligned}$$

At this point we are a bit stuck and it is helpful to draw a diagram. To do this observe that

$$F\,f\,(\alpha\,\mathrm{id}) = (F\,f \circ \alpha)\,\mathrm{id}$$

We need to exploit that α is natural, that means that the following square commutes:

$$\begin{array}{ccc}\underline{\mathbf{C}}(X, X) & \xrightarrow{\alpha X} & F\,X\\ {\scriptstyle \underline{\mathbf{C}}(f,X)}\downarrow & & \downarrow{\scriptstyle F\,f}\\ \underline{\mathbf{C}}(Y, X) & \xrightarrow[\alpha Y]{} & F\,Y\end{array}$$

Hence using naturality we can continue:

$$\begin{aligned}(F\,f \circ \alpha)\,\mathrm{id} &= (\alpha \circ \underline{\mathbf{C}}(f, X))\,\mathrm{id}\\ &= \alpha\,(\underline{\mathbf{C}}(f, X)\,\mathrm{id})\\ &= \alpha\,(\mathrm{id} \circ f)\\ &= \alpha\,f\end{aligned}$$

Hence $(\phi \circ \psi)\,\alpha\,f = \alpha\,f$ and using extensionality we can conclude $\phi \circ \psi = \mathrm{id}$ so together with $\psi \circ \phi = \mathrm{id}$ we have established that they constitute a natural isomorphism (we have already checked the ϕ is natural which is sufficient).

.

An important corollary of the Yoneda lemma is that the Yoneda embedding is full and faithful, that is the effect of the functor on morphisms is both injective (faithful) and surjective (full) which is a good reason why it deserves the name *embedding*. We can see this by applying the Yoneda lemma to $Y\ Z$

$$\begin{aligned}\phi : \int_{X:\underline{\mathbf{C}}} (Y\ Z)\ X &\cong \int_{Y:\underline{\mathbf{C}}} (\mathbf{Y}\ X)\ Y \to (\mathbf{Y}\ Z)\ Y \\ : \int_{X:\underline{\mathbf{C}}} \underline{\mathbf{C}}(X, Z) &\cong \int_{Y:\underline{\mathbf{C}}} (\mathbf{Y}\ X)\ Y \to (\mathbf{Y}\ Z)\ Y\end{aligned}$$

this looks like the morphism part of $\mathbf{Y}$ but is it? We need to do a little calculation:

$$\begin{aligned}\phi_X\ f\ g &= \mathbf{Y}\ Z\ g\ f \\ &= \underline{\mathbf{C}}(g, Z)\ f \\ &= g \circ f \\ &= \mathbf{Y}\ f\ g\end{aligned}$$

and hence the effect of $\mathbf{Y}$ on morphisms is an isomorphism and $\mathbf{Y}$ is full and faithful.

We can expand the isomorphism further:

$$\begin{aligned}\mathbf{Y} : \int_{X:\underline{\mathbf{C}}} \underline{\mathbf{C}}(X, Z) &\cong \int_{Y:\underline{\mathbf{C}}} (\mathbf{Y}\ X)\ Y \to (\mathbf{Y}\ Z)\ Y \\ : \int_{X:\underline{\mathbf{C}}} \underline{\mathbf{C}}(X, Z) &\cong \int_{Y:\underline{\mathbf{C}}} \underline{\mathbf{C}}(Y, X) \to \underline{\mathbf{C}}(Y, Z)\end{aligned}$$

We know that every functor preserves isomorphisms, a functor which is full and faithful also *reflects* them. That is if $Y\ f$ is an isomorphism then f is one.

Exercise 27 *Proof that every full and faithful functor reflects isomorphisms.*

This gives rise to a useful corollary: to show that $f : \underline{\mathbf{C}}(A, B)$ is an isomorphism it is enough to show that $\mathbf{Y}\ A$ and $\mathbf{Y}\ B$ are naturally isomorphic, i.e.

$$\int_{X:\underline{\mathbf{C}}} \underline{\mathbf{C}}(A, X) \cong \underline{\mathbf{C}}(B, X),$$

and this is often easier because we can just calculate with natural isomorphisms.

Sometimes we need the Yoneda lemma for covariant functors but this is no problem we just have to replace $\underline{\mathbf{C}}$ by $\underline{\mathbf{C}}^{\mathrm{op}}$. Hence another way to show that $f : \underline{\mathbf{C}}(A, B)$ is an isomorphism is to show

$$\int_{X:\underline{\mathbf{C}}} \underline{\mathbf{C}}(X, A) \cong \underline{\mathbf{C}}(X, B).$$

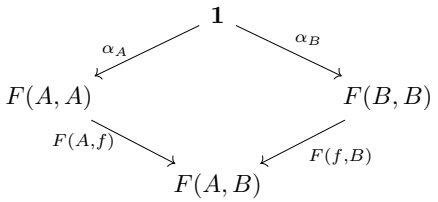
3.5 Ends and coends

Since I have introduced the notation $\int_{A:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(F\,A, G\,A)$ to define natural transformations, you are maybe wondering what the general meaning of these *integrals* are. It is clear that $\underline{\mathbf{D}}(F\,A, G\,A)$ is an function from $|\underline{\mathbf{C}}|$ but it is neither co- nor contravariant. However, it is a profunctor that is a functor of the type $F : \underline{\mathbf{C}}^{\mathrm{op}} \times \underline{\mathbf{C}} \to \underline{\mathbf{Set}}$ [12]

if we define it as $F(A^-, A^+) = \underline{\mathbf{D}}(F\,A^-, G\,A^+)$.

Exercise 28 *Fill in the morphism part of F.*

Now given a profunctor $F : \underline{\mathbf{C}}^{\mathrm{op}} \times \underline{\mathbf{C}} \to \mathbf{Set}$ we define its end $\int_{X:\underline{\mathbf{C}}} F(X, X)$ as a subset of the dependent function type $\alpha : \Pi_{X:|\underline{\mathbf{C}}|} F(X, X)$ such the following diagram commutes for any $f : \underline{\mathbf{C}}(A, B)$ in $\underline{\mathbf{Set}}$:



That is we define

$$\int_{X:\underline{\mathbf{C}}} F(X, X) = \{\alpha : \Pi_{X:|\underline{\mathbf{C}}|} F(X, X) \mid \forall f : \underline{\mathbf{C}}(A, B).F(A, f)\,\alpha_A) = F(f, B)\,\alpha_B\}$$

Exercise 29 *Show that instantiating the definition of ends given here gives rise to the definition of natural transformations given previously. That is $\int_{A:|\underline{\mathbf{C}}|} \underline{\mathbf{D}}(F\,A, G\,A)$ is the set of natural transformations.*

Dually we define coends $\int^{X:\underline{\mathbf{C}}} F(X, X)$ as a quotient of a Σ-type. That is we consider $\Sigma_{X:|\underline{\mathbf{C}}|} F(X, X)/\sim$ where for any $z : F(B, A)$ and $f : \underline{\mathbf{C}}(A, B)$ and $z : F(B, A)$, $(A, F(A, f)\,z) \sim (B, F(f, B)\,z)$. We can use coends to interpret the concept of an abstract datatype.

To remember when to use subscripts or superscripts we notice that in the case of ends the index is closer to the end of the page while for coends it is closer to the *coend*, i.e. the bginning, of the page.

[12] Here $\underline{\mathbf{C}} \times \underline{\mathbf{D}}$ is the product of categories: the objects are pairs of objects : $|\underline{\mathbf{C}} \times \underline{\mathbf{D}}| = |\underline{\mathbf{C}}| \times |\underline{\mathbf{D}}|$ and the morpshisms are pairs of morphisms: $\underline{\mathbf{C} \times \mathbf{D}}((C_0, D_0), (C_1, D_1)) = \underline{\mathbf{C}}(C_0, C_1) \times \underline{\mathbf{D}}(D_0, D_1)$.